Towards a fully MPEG-21 compliant adaptation engine: complementary description tools and architectural models*

Fernando López[1], José M. Martínez[1], Narciso García[2]
{*f.lopez,josem.martinez*}*@uam.es, narciso@gti.ssr.upm.es*
Grupo de Tratamiento de Imágenes
[1] Escuela Politécnica Superior. Universidad Autónoma de Madrid. E-28049. Spain.
[2] E. T. S. I. Telecomunicación. Universidad Politécnica de Madrid. E-28040. Madrid. Spain

Abstract. This paper develops a number of aspects of metadata-driven adaptation within the MPEG-21 framework that we consider are not clearly covered in the standard. Specifically, we introduce three complementary description tools that we have made use of and that we consider would fill certain gaps that we have identified in the standard. Besides, we offer a group of architectural design patterns along with a pragmatic group of considerations that we have gathered during the development of our MPEG-21 compliant adaptation engine.

Keywords: multimedia, adaptation, mpeg-21, description tools.

1. Introduction

MPEG-21 [1] is a multimedia framework proposed by the ISO (International Organization for Standardization) to specify a complete set of *description tools* that pave the ground for end-to-end solutions in multimedia systems. Within this framework, multimedia elements are represented using DIs (Digital Items) [2]. According to this framework a DI can convey *resources* (media files) and *descriptors* (metadata). The DI is a generic container capable of representing a great variety of multimedia. MPEG-21 Part 7 [3] standardizes a group of *description tools* capable of collecting the information that an adaptation engine can draw on to drive the adaptation of those DIs.

During the last years we have been working on the development of an adaptation engine named CAIN (Content Adaptation INtegrator) [4]. Numerous aspects have been elucidated since the first implementation. The implementation that we discuss in this document complies more properly and brings into play a wider range of description tools proposed by the MPEG-21 framework. Throughout its implementation we have encountered a number of important issues. We consider helpful to highlight

* Work supported by the European Commission (IST-IP-001765 – aceMedia, IST-FP6-027685 – MESH), Spanish Government (TEC2007-65400 - SemanticVideo) and Comunidad de Madrid (S-0505/TIC-0223 - ProMultiDis-CM). The Ministerio de Educación y Ciencia of the Spanish Government by means of the FPU grant issued to of the first author has also supported this work.

these difficulties for people involved in the construction of multimedia adaptation systems, especially if they are determined to provide MPEG-21 interfaces to their users. The clarification of these issues may also be useful for people who intend further interaction with other MPEG-21 compliant multimedia systems.

Section 2 revises the state of the art regarding multimedia automatic adaptation taxonomies and techniques. Section 3 introduces CAIN. Section 4 offers three new description tools that we consider would fill a few gaps that we have identified in the standard. Section 5 describes the architectural models that we have employed during the development of CAIN. Following the architecture of CAIN, in section 6 we motivate the usefulness of describing its adaptation capabilities with another group of description tools. Finally, section 7 provides the conclusions obtained from the development of CAIN. We expect that the ideas gathered in this document would be helpful for future advances in multimedia systems within the MPEG-21 framework.

2. State of the art

Nowadays a great deal of effort is being put into developing effective techniques to automatically adapt multimedia content to the usage environment. Roughly we can divide the automatic adaptation problem into two parts: first, the decision of which adaptation to perform and second, the execution of the selected adaptation. However the proposed techniques in the literature widely vary in nature, features under consideration and modus operandi. Certain techniques take into account a description of the environment [5][6], and model the adaptation decision problem making use of techniques like constraints satisfaction [5] or automatic planning [6]. Other techniques are concentrated on analysing the resource to measure the utility or the quality of the adaptation [7]. Usually there exists a distinction between mandatory constraints and user's preferences [8].

Most of the authors have made use of MPEG-21 Part 7 UED (Usage Environment Description) [3] to represent the features of the usage environment. Due to the all-purpose nature of MPEG-21 descriptions, we have selected this technology to investigate the multimedia decision problem. However, since we are only choosing a set of features, the selected features – described by means of the MPEG-21 description tools in this work – could be described without difficulty using another description technology. Thus, the ideas proposed in this document could be easily settled in adaptation engines that are focused on different base technologies.

2.1 Multimedia composition and adaptation levels

Multimedia content is typically made up of resource files where the content can be represented in different modalities (video, audio, images, ...). In general, those resources can be represented at two levels:

1 *Media level*, where a standardization body defines the whole format of the single modality media, and usually this media is stored in only one file (e.g. .jpg, .mp3).

2 *System level* (also named *multimedia level* or *application level* [11]), where the resource is made up of one or more media resources (e.g. .mpg, .html). Within the system level compositions, we can identify three important kinds of compositions: (a) *System level structure*, where the multimedia resource comprises several media and multimedia resources; (b) *System level layout*, where the spatial arrangement of the resources in the scene is portrayed (e.g. html); and (c) *System level synchronization*, where media and multimedia resources are temporally synchronized (e.g. MPEG audio and video streams, SMIL temporisation, Flash timeline).

The previous taxonomy gives rise to four kinds of adaptations:

1 *Media level adaptation*. This kind of adaptation is concentrated on transcoding (changes in the coding format, e.g., from .mp3 to .acc), transforming (changes in the coding parameters, e.g., frame size adjustment, frame dropping, scalable coding or summarizing media resources). There could be a combination of these two types in a single operation, that is, changes at the same time coding format and parameters). In addition, the media level adaptation could implement transmoding between media modalities (e.g. video to images slideshow).
2 *Structure level adaptation*. This kind of adaptation is concentrated on adapting a resource that is the union of (or has references to) other media resources (e.g. DI, BIFS, HTML).
3 *Layout level adaptation*. This kind of adaptation is concentrated on changing the arrangements of the constituent elements in the scene (e.g. .html, .smil).
4 *Synchronization level adaptation*. This kind of adaptation is concentrated on modifying the timeline of the constituent resources (e.g. .mpg audiovisual files summarization, or SMIL images serialization).

2.2 DIA Description Tools

In this section we are going to talk about two *DIA (Digital Item Adaptation) Description Tools* proposed in MPEG-21 Part 7. In section 4.2 we will introduce an alternative *DIA Description Tool* that we propose to be useful in order to drive the adaptation, and we will illustrate a suitable use case where this new *DIA Description* is thoroughly applicable.

Usage Environment Description Tool

The MPEG-21 Part-7 [3] standard has defined, among others, the *Usage Environment Description* (UED) *Tools*, which is a set of description tools widely used to describe the target environment of the adaptation. Instances of the *UED Tools* are referred as *UED Descriptions*, and with regard to the UED there exist four main *UED Descriptions*: *Terminals*, *Networks*, *Users* and *NaturalEnvironments*. Since the standard allows each *UED Description* to be instantiated from zero to *n* times, the UED acts as a database where the features of different usage environments are described.

DIA Configurarion Tool

The *DIA Configuration* (DIAC) *Tool* [3] is a tool that allows guiding the adaptation considering the DI author's intentions. This information is conveyed at DI level in a *Descriptor* element. Specifically two methods to provide DIAC information are available. The first method is through the use of the *UserSelection* /*BackgroundConfiguration* elements, which allows the DI's author to suggest the means by which *Choice*/*Selection* of the DI should be processed. The second method is through the use of the *SuggestedDIADescriptions* used by the author of the DI to provide XPath [12] expressions pointing to the fragments of the *DIA Description* to be used during the adaptation.

The *DIAC Tool* proposes the existence of a negotiation protocol between the DI server and DI client. When the DI client requests a DI to the DI server, the DI server sends a DI along with a *DIAC Description* to the DI client. The *DIAC Description* can be used to decide where to perform the adaptation: client, proxy and server. A combination of them is also possible.

Comentario [flh1]:

3. Introduction to CAIN

CAIN [4], our MPEG-21 adaptation engine implementation, is designed to perform all the levels of adaptations described in section 2.1. Since CAIN is a metadata driven adaptation engine, it makes use of annotations on the resources to drive the adaptations. Specifically, multimedia content is annotated using the MPEG-7 [9] standard, and the adaptation system makes use of the MPEG-21 framework description [1]. The inputs and outputs of CAIN are represented as MPEG-21 DIs (Digital Items) [2]. The main reasons that justified our selection of MPEG-21 DIs is that (1) they are capable of representing virtually every kind of media or system level composition and (2) that they allow for the annotation of the multimedia resources. In particular: media level adaptation can be performed referencing a media resource in the *Resource* element of its *Component* element. Moreover, this *Component* may include a *Descriptor* element with metadata about the resource. In the case of structure level adaptation the *Component* elements of the DI can be modified as needed.

Fig. 1 shows an example of a DI that includes several *Component* elements. In this figure, there exists a *Component* with a HTML web page, which includes references to video and image resources in other *Component* elements. According to the taxonomy given in section 2.1, the image is a media level resource, the video is a synchronization level resource, and the HTML page is a system level structure composition that has references to the video and image resources. In this scheme, when the DI is adapted, both the web page and the linked resources are adapted. In the case of layout adaptation an HTML or SMIL file is conveyed in the *Descriptor* of the *Component* of the DI. The usage environment restrictions (such as screen size) are used to perform the layout level adaptation. Synchronized level adaptation can be applied to transcoding or transforming multimedia level synchronized resources (e.g. .mpg files).

As CAIN is designed to be capable of adapting every kind of media and multimedia document, hereafter we will use the term *resource* to refer to the file referenced by the

Resource element. This resource may be a media resource (e.g. .jpg) or a multimedia resource (e.g. .mpg or .html).

We will use the term *DI adaptation* to refer to the type of adaptation performed by CAIN over a DI. The input to a DI level adaptation is a DI, and the output of the adaptation process is represented as an *adapted DI*. As a DI could comprise one or more *Component* elements, we have decided that adaptation within CAIN is going to be performed at *Component* level by software tools named *Component Adaptation Tools* (CATs). We also have considered dependencies in situations like the one plotted in Fig. 1. In this situation the resource of a *Component* (the HTML page) includes references to the resources of other *Component*s.

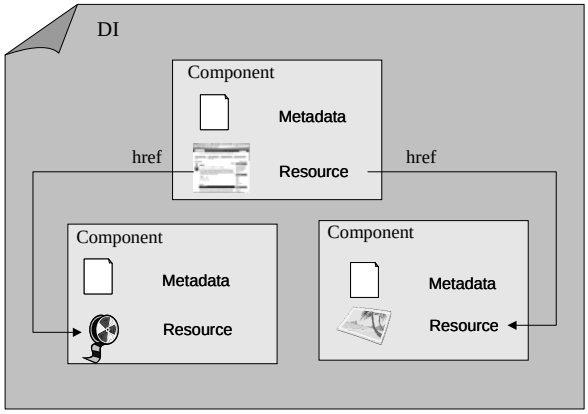

Fig. 1: DI composition within the CAIN

Metadata may be automatically extracted by the CATs, or may be provided as input by the *Components*. Usually *Component* level adaptation is performed in two steps: During the first step the resource is retrieved from the media repository. In addition, if semantic adaptation is used, the resource and associated metadata may be analysed. During the second step metadata conveyed in the *Component* and the information that results from analysing the content is used to adapt (transcode, transform or transmode) the resource and associated metadata.

4. Describing the configuration of the adaptation engine

This section takes on the rationalization of the solution that we suggest for reusing the MPEG-21 description of the content and context onto different multimedia adaptation requests. The upshot of this discussion is in section 4.4 where we will offer the use of an MPEG-21 complementary DIA *Description Tool*.

4.1 Content DI, Context DI and Configuration DI

Authors (for instance [10]) have modelled their adaptation engines assuming the existence of two kinds of DIs: the Content DI and the Context DI. The *Content DI* is a DI that carries out the multimedia document as well as the DIAC. The *Context DI* contains a description of the UED.

We consider very useful the exploitation of the idea behind the *DIAC Tool* whenever an adaptation is going to be performed. This is because (as indicated in section 2.2) the UED allows the description of multiple usage environments, and hence it is important to know which environment is going to be the target of the adaptation. Although there exist scenarios where the DIAC mechanism is applicable, we have identified two drawbacks in the DIAC mechanism: (1) there exists a coupling between the Content DI (with the multimedia document and DIAC) and the Context DI (with the UED). That is, the Content DI encloses the alternative adaptation options to the usage environment, and as a consequence, the Content DI document must be modified when the usage environment capabilities (Context DI) change. (2) The DIAC mechanism assumes that the possible usage environments are known when the Content DI is created.

We propose that in order to avoid the first drawback it is useful to make use of three DIs: the *Content DI* (with the multimedia document), the *Context DI* (that acts as a database where the different terminals, networks, users and natural environments under consideration are described), and the *Configuration DI* that encloses the DIAC. The exact way to represent this configuration is the innovative part of this section. In the next section we are going to rationale it. The main aim of our proposal is that the Content DI will not be modified when the usage environment (located in the Context DI) changes. The Configuration DI also solves the second drawback: the Content DI and Context DI could be created and stored into the MPEG-21 multimedia system during the adaptation engine implementation. The Configuration DI would be created only when the adaptation engine has been deployed and is ready to start serving adaptation requests.

4.2 The Adaptation Request Configuration description tool

Currently the MPEG-21 Part 7 standard proposes two different *DIAC Descriptions*: (1) The *UserSelection*/*BackgroundConfiguration* elements that indicate whether the DI *Choice*/*Selection* mechanism must be presented to the user or automatically decided by the system, and (2) the *SuggestedDIADescriptions* where the DI's author suggests which *DIA Descriptions* should be used to make decisions. As indicated in section 2.2, both *DIAC Descriptions* assume the existence of a negotiation mechanism and place the *DIAC Descriptions* in the DI that is going to be consumed. We propose that under certain circumstances (for example when the adaptation engine is implemented as a middleware that provide an API instead of as a network agent) it is useful to relax the previous assumptions and make use of a third (no considered in MPEG-21) *DIAC Description Tool* that we will refer as *Adaptation Request Configuration* (ARC) *Description Tool*. The *ARC Description Tool* is in charge of selecting a zero or

one instance value of the *Terminals*, *Networks*, *Users* and *NaturalEnvironments* available in the UED. For example, one adaptation request could be intended to adapt one Content DI to a mobile *Terminal* and another adaptation request could be intended to adapt the same Content DI to a laptop *Terminal*. Note that the Content DI and Context DI could be created before deploying the adaptation engine in a multimedia system. However the set of feasible *ARC Descriptions* depends on the *UED Description* of the multimedia system where the adaptation engine has been deployed, and therefore the *ARC Description* will not be created until the adaptation engine is deployed in a particular multimedia system.

Comentario [jmms2]: No me gusta mucho tanta sigla en el título de la sección pero….

4.3 DIs and UED in CAIN

In this section we introduce the way in which the Content DI and Context DI are implemented in CAIN. When CAIN is deployed in a multimedia system, we presuppose the existence of multimedia represented as a group of Content DIs. In section 5 we will discuss that CAIN is also capable of representing non-MPEG-21 multimedia content as DIs before performing the adaptation. For example, *video1.xml*[1] is a Content DI with only one *Component* element that includes a *Resource* element that points out to a video resource and a *Descriptor* element where the resource is annotated in MPEG-7.

An example of a Context DI can be found in the *ued.xml* file. The document contains the *Terminals*, *Networks* and *Users* elements, which have an associated unique ID. These IDs will be used in the next section to indicate which *Terminal*, *Network* and *User* to must be used during the adaptation.

4.4 ARC driven adaptation example in CAIN

Before a DI level adaptation could be executed in CAIN, an *ARC Description* must be provided. If the *ARC Description* is not specified, CAIN is not capable of adapting the Content DI, because there is no way to decide the transformations to perform over the elements of the Content DI. Listing 1 shows an example of an *ARC Description*.

```
<?xml version="1.0" ?>

<DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Description xsi:type="ARCType">
  <Terminal>
   <Id>mobile_1</Id>
   <OperationMode>
   //TerminalCapability[@type='dia:DisplaysType']/Display/
   DisplayCapability[@type='dia:DisplayCapabilityType']/Mode/
   Resolution/[@id="176x144"]
   </OperationMode>
```

[1] Due to space constraints, at http://www-gti.ii.uam.es/publications/dt_arch/ we provide a number of documents online

```
  </Terminal>
  <Network>
   <Id>ethernet</Id>
  </Network>
  <User>
   <Id>reporter</Id>
  </User>
 </Description>
</DIA>
```

Listing 1: *ARC Description* example

Comentario [jmms3]: Habría qe intentar que los listing no se corten entre páginas…

Within CAIN, as the target *Terminal* description contains basic information (such as *CodecCapabilities*, *Display*, ...) essential to decide the adaptation that should be performed, we have marked the *Terminal* element as mandatory (see *carc.xsd*). In Listing 1 the *Terminal* with ID *mobile_1* has been selected. The description of this *Terminal* is located in the *ued.xml* document. The two other *UED Descriptions* that CAIN supports (*Network* and *User*) have been left optional (see *carc.xsd*), although they are useful to reach a better adaptation. The *Network* and *User* elements of Listing 1 are optional and, if present, they would contain respectively the ID of the *Network* and *User* of the Context DI to use during the adaptation request. CAIN does not make, currently, use of the *NaturalEnvironment* descriptions.

4.5 Operation modes

In addition to the ID of the *UED Descriptions* to use during the adaptation, the *ARC Description* is able to describe what we refer to as *operation modes*. An *operation mode* allows selecting parts of the *UED Description* whenever several modes could be used. For instance, a *Terminal* might support different screen resolution modes. In this case the adaptation engine must be capable of selecting one of these operation modes, but additionally the *ARC Description* can be used to signal a specific operation mode that the adaptation engine must obey.

5. Modular architecture of CAIN

This section provides the architectural model of CAIN. The purpose of this section is twofold: (1) to suggest a group of software design patterns that may result useful to future adaptation engine implementors, and (2) to explain the applicability of the notion of CATs (introduced in section 2.1), which are reusable software tools that could be put into operation in different adaptation scenarios. After those concepts will be clarified, section 6 presents two description tools that we consider helpful to make adaptation decisions.

5.1 CAIN modules

CAIN serves requests through two external programming interfaces: (1) *The media level transcoding interface* is concentrated on performing blind adaptation of a media resource. In addition to media level, this interface is also capable of performing system level adaptation of videos composed of one or more audio and visual streams. (2) *The DI level interface* is concentrated on performing system level (semantic or blind) adaptations where metadata is used for helping during adaptation.

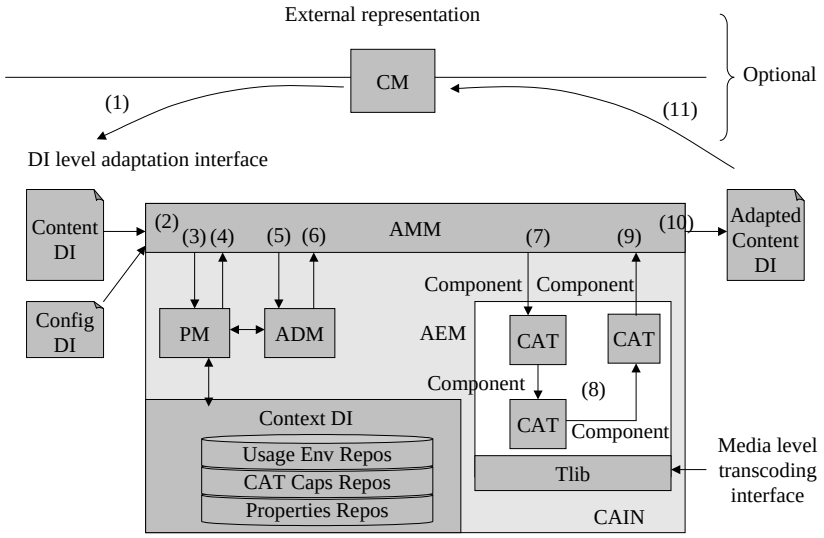

Fig. 2: Control flow within CAIN

The CAIN engine has been functionally divided into modules. These modules and the control flow along the adaptation process are depicted in Fig. 2. The media level transforming interface is served through a conventional software library that we have implemented named *Tlib*. This library implements blind image transcoding and transforming, video transcoding and transforming, video truncating and video frame extraction. The CATs are software modules that perform (semantic or blind) *Component* level adaptation. As each *Component* element includes both the resource and its metadata, the CATs serve parts of the DI level interface. The CATs take advantage of the *Tlib* to perform parts of the blind adaptation of the resources referenced in the *Components*. In addition, the CATs adapt the metadata of the *Component* elements. Note that whereas CATs are capable of performing (semantic or blind) *Component* transcoding, transforming, transmoding and summarization adaptations, the *Tlib* is concentrated on achieving blind media level operations (e.g. transcoding an MPEG-2 MP@ML video to MP4 video container with AAC/H264 streams).

Comentario [jmms4]: Qué quieres decir con “servir”?

Comentario [jmms5]: AAC/H264 es audio/video, mientras que MP@ML es solamente video. Habría que indicar un formato de audio –en TVD puede ser Mpeg-2 audio en DVB, o Dolby AC3 en ATSC y DVD-)

The Context DI encompasses a number of repositories. The *Usage Env Repository* is a set of documents annotating the usage environments using the *UED Tools* (introduced in section 2.2). The *CAT Capabilities Repository* is another set of documents annotating the *conversions* that the available CATs are capable of performing. Each conversion has a set of valid input and output properties along with its corresponding values. Every CAT must have an associated *CAT Capabilities* document where the different conversions that the CAT is capable of accomplishing are described. The *Properties Repository* is yet another set of documents committed to relate the CAT Capabilities and the UED. These relationships can be represented as properties that take one or more values. Specifically, we use XPath expressions to represent the values of those properties. In addition, it's important to remark that there must exist one CAT Capabilities file per CAT, but the Properties Repository stores properties that are common for the CATs altogether, and in this sense, all the CAT implementers must provide these properties (e.g. *media_system_format*, *visual_coding*, *visual_height*, ...). Some of these properties are marked as mandatory and the rest as optional.

The *Parsing Module* (PM) is the module in charge of loading and querying the Content DI, Configuration DI and Context DI. Within the PM we also find the *Coupling Module* (CM). There exist a wide variety of multimedia representation standards and systems. CAIN is a software engine that is designed with the ability to be integrated in heterogeneous environments. The CM is the module interfacing with other components that may be using *external technology* (i.e., non-MPEG-21 technology) to represent multimedia (e.g. HTML, SMIL, NewsML). The main purpose of the CM is to enable the integration of CAIN adaptation services into those heterogeneous multimedia systems. The CM is in charge of transforming that external representation of multimedia onto an MPEG-21 compliant input DI that CAIN is capable of processing. In addition, the CM is in charge of transforming the adapted output DI onto its external representation.

When a DI arrives, the *Adaptation Decision Module* (ADM) is in charge of deciding the *sequence of conversions* committed to transforming the initial Content DI onto an adapted Content DI taking into account the information in the Configuration DI and Context DI. Different CATs sequentially perform the conversions of the sequence. A CAT receives as input a Content DI and generates as output another Content DI so that a conversion operation has been performed over one or more of its *Components*. The module in charge of coordinating the CATs execution is the Adaptation Execution Module (AEM).

The *Adaptation Management Module* (AMM) is in charge of coordinating the whole Content DI level adaptation process. Modules below the AMM are concentrated on performing different tasks requested by the AMM. It's important to remark that, although the input/output unit of the DI level interface is the Content DI, the adaptations are always performed over one or more *Components* of the Content DI, and not over the entire Content DI. In this sense the DI acts as a container of *Components*. Before adaptation could be performed, CATs must be installed in the AEM and its conversion capabilities registered in a *CAT Capabilities Repository*. Besides, information about terminal capabilities, network descriptions, user characteristics and preferences must be registered in the *Usage Environment Repository*.

5.2 Control flow

Numbers in Fig. 2 provide the temporal order (control flow) of the tasks involved in the adaptation process. (1) The CM is the module in charge of dealing with the specific environment where CAIN has been deployed in order to provide its multimedia adaptation services. Different instances of the CM are interchangeable modules created to interact with the specific multimedia representation details of the deployment system. They transform the external multimedia representation of the environment onto MPEG-21 compliant DIs that CAIN is capable of processing. (2) When a Content DI along with a Configuration DI arrives (via the DI level adaptation interface), they are sent (3) to the PM to represent that information in memory objects. The PM is in charge of reading the *DIA Descriptions* (ARC, UED, CAT Capabilities and Properties). The information in those documents is cached to improve performance. (4) That information is returned to the AMM, and (5) the information is sent to the ADM that decides which conversions (represented as a sequence of conversions with the corresponding set of proper parameters) will be used to achieve the best adaptation. Specifically, the sequence of conversions returned by the ADM is the sequence of conversions that should be executed over the *Component* elements of the Content DI, and not over the entire Content DI. (6) The selected sequence of conversions is sent back to the AMM. If the decision returned by the ADM is that the adaptation cannot be performed (given the input Content DI, Configuration DI, available CATs and usage environment), the AMM must inform of that condition, and the adaptation process finishes. Assuming that the ADM returns a valid sequence of conversions, (7) this sequence is transmitted to the AEM, which executes the consecutive sequence of corresponding CATs. (8) During the CAT's execution, optionally, CATs may append information to the *Descriptor* element of the *Component* so that subsequently CATs could use it. We use the name static decisions to refer to the usage environment and user preferences based decisions, because they don't depend on the resource content (only the resource description), and the ADM can make those decisions. On the contrary, we use the term *dynamic decisions* to refer to adaptation decisions that perform measures over the resource content, and cannot be taken until the resource is available (such as utility and quality based decisions [7]). As explained in [13], CATs are non-deterministic since they can make those dynamic decisions. (9) When all the conversions of the sequence have been executed, the AEM returns the adapted DI to the AMM. (10) CAIN uses the DI level adaptation interface to return the adapted Content DI to the caller. (11) Eventually, the adapted Content DI may need to be transformed to an external representation. In this case the CM receives the adapted Content DI and transforms it into an external representation.

6. Description of the adaptation capabilities

In [14] we introduced the notion of CATs (see section 2.1) and its associated CAT Capabilities (see section 5). In this section we are going to provide a number of enhancements over the CAT Capabilities that we have identified and implemented into CAIN. In particular, there exist two main changes with respect to the CAT Capabili-

ties model previously proposed in [14]: (1) Each CAT can incorporate several *conversions*[2]. This is due to the fact that the conversion capabilities that can be performed are not always easy to describe if we do not break apart the capabilities of an individual CAT into several conversion capabilities elements. For example, if a specific CAT is capable of accepting JPEG and PNG images, but PNG images are accepted only in greyscale, whereas JPEG images are accepted in colour and greyscale. In this case the CAT capabilities must be split into two separate conversion capabilities: one conversion capabilities exposing that PNG images are accepted in greyscale and another conversion capabilities exposing that JPEG images are accepted in both colour and greyscale. (2) As explained in [13], within CAIN the *potential adaptation capabilities* are represented as multivaluated properties, i.e., properties that can take several possible values (e.g. *format*={*mpeg*-1, *mpeg*-2, *mpeg*-4}). We have modified the description of the conversions so that each input and output property could take multiple values.

```
<?xml version="1.0" ?>
<dia:DIA xmlns="urn:gti:cain-cat-capabilities"
         xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <dia:Description xsi:type="CATCapabilitiesType"
                          id="video_transcoder_cat">
  <!-- Offline conversion using the ffmpeg command -->
  <ConversionCapability xsi:type="ConversionCapabilityType"
                                      id="mpeg4_offline_transcoder">
   <Preconditions>
    <URL>
     <AnyValue/>
    </URL>
    <MIMEType>
     <ValueSet>
      <Value href="video/mpeg">MPEG video</Value>
      <Value href="video/mpg">MPEG video</Value>
      <Value href="video/avi">AVI video</Value>
     </ValueSet>
    </MIMEType>
    <Bitrate>
     <RangeValueSet from="40000" to="2000000"/>
    </Bitrate>
   ..............................
   </Preconditions>
   <Postconditions>
    <URL>
     <AnyValue/>
    </URL>
    ..............................
   </Postconditions>
  <!-- Online conversion using the ffmpeg libraries and JNI -->
  <ConversionCapability xsi:type="ConversionCapabilityType"
                                      id="mpeg4_online_transcoder">
  ................................
  </ConversionCapability>
 </dia:Description>
</dia:DIA>
```

Listing 2: *CAT Capabilities Description* example

Comentario [jmms6]: Idem de no partir

[2] This term has been selected in accordance with the term *Conversion Tool* that MPEG-21 Part 7 Amendment 1 proposes to represent the software or hardware element that performs the adaptation.

The CAT Capabilities Tool that we propose is available online in *ccatc.xsd*. The *CATCapabilitiesType* type represents a particular CAT, and the *ConversionCapabilityType* type represents each conversion that the CAT is capable of performing. It's worth to notice that the *CATCapabilitiesType* is defined as a derivation by restriction of the MPEG-21 *DIADescriptionType*, and hence the *CATCapabilitiesType* can be seen as a non-MPEG-21 standardized DIA *Description Tool*. For instance, Listing 2 shows the more relevant parts of a video transcoding CAT example available online in *video_transcoding_cat.xml*.

This CAT comprises two *ConversionCapability* elements named *mpeg4_offline_transcoder* and *mpeg_online_transcoder*. The *Preconditions* and *Postconditions* elements convey information related to the media formats that the conversion respectively accepts and produces. Those inputs and outputs correspond with the preconditions and postconditions of the conversion according to the model that we have proposed in [13].

6.1 The Properties Repository description tool

In early versions of CAIN, the PM was in charge of parsing the different DIs. For each document, it produced a hierarchical tree of objects. After that the ADM implementation went through those hierarchies searching for the specific values needed to make decisions about the adaptations to perform. This course of actions proved tedious to implement and difficult to maintain since changing one element of an individual document implied searching for the use of this property in the algorithm of the ADM and updating the algorithm in one or more points.

Those difficulties motivated the idea of gathering the relevant properties to be used by the ADM in a Properties Repository. Each property is represented as a label with an associated XPath expression that points out to the specific part of the DIs or CAT Capabilities where their values are located. Even though the PM is still in charge of parsing the documents and loading them in memory as hierarchies of objects, the access to the properties is no longer performed directly by the ADM. Instead, those values are accessed by means of properties stored into the Properties Repository. In this way, modifications in the location or in the number of the properties are done by changing the Properties Repository, instead of changing the ADM source code.

The Properties Repository scheme that we propose is available online in *cpr.xsd*. The *PropertiesReposType* type is also defined as a derivation by restriction of the MPEG-21 standard *DIADescriptionType* and includes four main elements: *ResourceProperties*, *CATProperties*, *ConversionProperties* and *UsageEnvProperties*, which correspond to the four main groups of properties that we have identified. Those groups are further discussed in [13]. Each one of those elements contains XPath expressions that point out to the specific keys and values of the properties within the MPEG-7, MPEG-21 and CAT Capabilities description documents. Listing 3 shows the more relevant parts of a *Properties Repository* example (available online in *pr.xml*). This document contains all the XPath expressions in use currently within CAIN.

```
<dia:DIA xmlns="urn:gti:cain-properties-repos"
         xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
```

```
<dia:Description xsi:type="PropertiesReposType">
 <ResourceProperties required="true">
  <Property name="id" required="true"  xpath="/@id"/>
  ..........................................
 </ResourceProperties>
 <CATProperties required="true">
 ..........................................
 </CATProperties>
 <ConversionProperties required="true">
  <Property name="lossy" required="false"  xpath="/Lossy"/>
  ..........................................
 </ConversionProperties>
 <UsageEnvProperties required="true">
 ..........................................
 </UsageEnvProperties>
</dia:Description>
</dia:DIA>
```

Listing 3: *Properties Repository Description* example

This set of properties can be used in different adaptation scenarios where different DIs, CAT Capabilities and UED documents (that comply with the prearranged schema) can be used without making changes in the ADM or AEM algorithms.

7. Conclusions

In this document we propose the distinction between the *UED* - capable of describing different usage environments - and the *ARC Description* - that selects one of those environments before performing the adaptation -. In addition, we have emphasized the different kinds of multimedia composition and adaptation levels that ought to be contemplated by an all-purpose adaptation engine. CAIN is a multimedia adaptation engine that can be integrated within large-scale multimedia systems with the purpose of providing multimedia adaptations services. It has been successfully put into practice in the aceMedia and MESH projects that are referenced in the acknowledgments of this paper. In order to deal with the diversity, CAIN proposes two instruments: (1) the wide variety of multimedia description formats that can be dealt with the use of the CM (see section 5.1), so that the external multimedia representation is transformed to MPEG-21 DIs before performing the adaptation. The external representation could be recovered again after adaptation. (2) The large quantity of multimedia conversions that could be used to adapt media makes it unfeasible to implement all of them, so we propose the use of pluggable CATs. Their functionality is described in the CATs Capabilities, so that they can be used as soon as they are installed in the adaptation engine. Since each CAT could be capable of performing several conversions, we propose to divide the *CAT Capabilities Description* into *Conversion Descriptions*. In addition, in section 6.1 we proposed the use of a *Properties Repository* that contains the set of properties that are going to be used by the ADM to make decisions. Assuming that those *CAT Capabilities Repository* and *Properties Repository* are provided, we have demonstrated that the automatic decision of the adaptation to perform can be addressed more effectively and efficiently.

8. References

[1] Ian S. Burnett, F. Pereira, R. V. de Walle, R. Koenen, *The MPEG-21 Book*, John Wiley and Sons. 2006.

[2] 'MPEG-21 Part 2: Digital Item Description', TR, ISO/IEC 21000-2. 2005.

[3] 'MPEG-21 Part 7: Digital Item Adaptation', TR, ISO/IEC 21000-7. 2004.

[4] J.M. Martínez, V. Valdés, J. Bescós, L. Herranz, 'Introducing CAIN: A metadata-driven content adaptation manager integrating heterogeneous content adaptation tools'. Proceedings of WIAMIS'05, Montreux, France. 2005

[5] F. López, J. M. Martinez, 'Multimedia Content Adaptation Modelled as a Constraints Matching Problem with Optimisation' in Proceedings of the International Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS'07, pags 82-85 (ISBN 0-7695-2818-X), Santorini, Greece. June 2007.

[6] D. Jannach, K. Leopold, Ch. Timmerer, H. Hellwagner, 'A knowledge-based framework for multimedia adaptation', *International Journal on Applied Intelligence, Springer, 24* **2**, 109:125. April 2006.

[7] M. Prangl, T. Szkaliczki, and H. Hellwagner, 'A Framework for Utility-Based Multimedia Adaptation', *IEEE Transactions on Circuits and Systems for Video Technology* **17**, 719-728. June, 2007.

[8] B. Köhncke, W. T. Balke, 'Preference-driven personalization for flexible digital item adaptation', *Multimedia Systems* **13, 2**, 119-130. August, 2007.

[9] 'MPEG-7 Part 5 Multimedia Content description Interface', Technical report, ISO/IEC 15938. 2003.

[10] Saar De Zutter, Rik Van de Walle, "Enhanced Quality of Experience in Heterogeneous Environments from a Content Author's Perspective". Sixth FirW PhD Symposium, Ghent University, 30th November 2005.

[11] Mazen Almaoui, Azadeh Kushki, Konstantinos N. Plataniotis, 'Metadata-driven multimedia transcoding for distance learning', *Springer Berlin / Heidelberg* **12**(6), 505-520. 2007.

[12] 'XML Path Language (XPath) Version 1.0', Technical report, WWW Consortium (W3C). November 1999.

[13] F. López, D. Jannach, J. M. Martínez, Christian Timmerer, Hermann Hellwagner, Narciso García, 'Multimedia Adaptation Decisions Modelled as Non-Deterministic Operations' WIAMIS 2008, Klagenfurt. May 2008.

[14] V. Valdés, J.M. Martínez, 'Content Adaptation Tools in the CAIN framework' in Visual Content Processing and Representation, L. Atzori, D.D. Giusto, R. Leonardi, F. Pereira (eds.), Lecture Notes in Computer Science, Vol. 3893, Springer Verlag, pp. 9-15. 2006.